REMARKS

Claims 1 through 26 are pending.

Claims 1 through 11 have been rejected under 35 U.S.C. § 101.

Claims 1 through 26 have been rejected under 35 U.S.C. § 103 (a).

Rejection under 35 U.S.C. § 101

Examiner has rejected claims 1 through 11 under 35 U.S.C. § 101, asserting that the claimed invention is directed to non-statutory subject matter. Applicant traverses the rejection and respectfully requests reconsideration.

As interpreted by the Federal courts, 35 U.S.C. § 101 has two purposes. First, 35 U.S.C. 101 defines which categories of inventions are eligible for patent protection. Second, 35 U.S.C. § 101 serves to ensure that patents are granted on only those inventions that are "useful". Thus to satisfy the requirements of 35 U.S.C. § 101, an applicant must claim an invention that is statutory subject matter and must show that the claimed invention is "useful" for some purpose either explicitly or implicitly. See MPEP 2107.01.

Examiner has asserted that claims 1 through 11 are not directed to statutory subject matter. Applicant respectfully disagrees.

Claims 1 through 11 are directed to "a server computer system". A server computer system is clearly a machine patentable under 35 U.S.C. 101. Therefore, claims 1 through 11 clearly satisfy the first criterion of 35 U.S.C. § 101.

Record-keeping machine systems are clearly within the "technological arts." Such machine systems, which comprise programmed digital computers,

are statutory subject matter under the provisions of 35 U.S.C. § 101, and "claims defining them must be judged for patentability in light of the prior art." *In re Johnston*, 502 F.2d 765, 183 U.S.P.Q. 172, 183, 184 (C.C.P.A. 1974). A computer operating pursuant to software may represent patentable subject matter, provided that the claimed subject matter meets all of the other requirements of Title 35 (U.S.C.) *In re Alappat*, 33 F.3d 1526, 31 U.S.P.Q.2d 1545, 1558 (Fed. Cir.1994).

In Examiner's explanation for the rejection, Examiner has asserted that claims 1 through 11 do not define any specific hardware. Therefore, Examiner has argued the system is not "tangible embodied in a manner so as to be executable".

Examiner appears to be referring to the criterion that a scientific principle, divorced from any tangible structure, can be rejected as not within the statutory classes. See *O'Reilly v. Morse* 56 U.S. (15 How.) 62 (1854). MPEP 706.03(a) discusses this criterion. However, this criterion is inapplicable to claims 1 through 11 because claims 1 through 11 are not claiming a scientific principle but are claiming a server computing system.

Applicant notes that when determining whether a claim fulfils the requirements of 35 U.S.C. § 101, it is not sufficient to evaluate the elements of the claim separately. Rather, when determining whether a claim fulfils the requirements of 35 U.S.C. § 101 it is necessary to consider the claim as a whole, including the preamble and all the elements. The preamble of each of

the claims 1 through 11 recites that a server computing system is being claimed. This is sufficient to satisfy the very low threshold for patentability under 35 U.S.C. § 101.

Rejections under 35 U.S.C. § 103(a).

Examiner has rejected claims 1 through 3, 6 through 9, 12 through 14, 16 through 18, 21, 23 and 24 under 35 U.S.C. § 103 (a) as being unpatentable over USPN 5,935,212 (Kalajan) in view of USPN 6,553,363 (Hoffman).

Examiner has rejected claims 4, 11, 20 and 26 under 35 U.S.C. § 103 (a) as being unpatentable over Kalajan in view of Hoffman in view of USPN 6,026,413 (Challenger).

Examiner has rejected claims 5, 12, 15, 19, 22 and 25 under 35 U.S.C. § 103 (a) as being unpatentable over Kalajan in view of Hoffman in view of USPAP 2004/0064515 (Hockey).

Applicant respectfully traverses the rejections. Below, Applicant sets out subject matter in each of the independent claims not disclosed or suggested by the cited art. In view of this, Applicant believes all the claims are patentable over the cited art.

Independent Claim 1:

Independent claim 1 sets out a server computing system that includes an application. The application includes a persistent process that generates dynamic and interactive hypertext markup language (HTML) content for the

application. This is not disclosed by Kalajan or any of the other art cited by Examiner in the Office Action mailed March 30, 2005.

In prior art systems, the standard method for generating dynamic content is to use the Common Gateway Interface (CGI). In this model, a user request for dynamic content arrives at a web server and is handled by execution of a specific program on the web server, usually in a protected memory space. The program then terminates after the dynamic content is generated. The use, by the present invention, of a persistent process that generates dynamic and interactive hypertext markup language (HTML) content is, for certain applications, a significant improvement over the prior art. None of the art cited discloses or suggests this improvement.

Kalajan discloses a persistent proxy socket application 56, shown in Figure 4. Persistent proxy socket application 56 handles primitives (or other messages) to and from TCP mail server 58. See Kalajan column 4, lines 12 through 14. Persistent proxy socket application 56 does not generate dynamic and interactive hypertext markup language (HTML) content for the application, as set out in claim 1 of the present application.

Kalajan is not concerned with generating HTML content. Rather, the whole thrust of Kalajan is to emulate a connection-oriented session across a network using a stateless communication protocol. See the Abstract of Kalajan. Thus, the persistent proxy socket application 56 disclosed by Kalajan is used specifically for the purpose of handling primitives (or other messages) to and from TCP mail server 58. Proxy socket application 56 remains active during an

entire emulated TCP session, handling TCP message flow to and from TCP mail server 58. This allows transient proxy socket application 54 to terminate upon each output message, as required by HTTP. See Kalajan at column 4, lines 16 through 28. Nowhere does Kalajan disclose or suggest that proxy socket application 56 generates dynamic and interactive hypertext markup language (HTML) content for an application.

Examiner has asserted that Kalajan discloses that CGI program on the server could call the proxy socket applications directly to implement an HTML based email client (col. 6, lines 46 -51). However, this statement by Kalajan does not specifically state where or how HTML content is generated. In known prior art systems, HTML content is generated by transient CGI programs, such as the CGI program mentioned by Kalajan. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan are any different than the transient CGI programs of the known prior art. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan include persistent processes. Nothing in Kalajan discloses or suggests that HTML content is generated anywhere else besides where HTML is typically generated, within transient CGI programs, such as the CGI programs mentioned by Kalajan.

Hoffman's discussion of persistent *data*:

Examiner has also asserted that Hoffman, at column 2, lines 29 through 53 and column 8, lines 38 through 49, discloses the concept of generating dynamic and interactive HTML content in which dynamic HTML pages may be created

using JavaScript to process user input and maintain persistent data using special objects, files, and relational databases (col. 2, lines 29-53, col. 8, lines 38-49). However, Hoffman gives no information about how HTML content is generated on a server. Rather, Hoffman concerns a method and apparatus for processing a document retrieved *from* a server. See the Abstract. The sections of Hoffman cited by Examiner (col. 2, lines 29-53, col. 8, lines 38-49) primarily discuss how a browser can interpret *client-side* JavaScript statements embedded in an HTML page received from a server. Hoffman gives no information on how the HTML page is generated.

At column 2, line 36 through 39, Hoffman indicates that using JavaScript, dynamic HTML pages may be created in which the pages process user input and maintain persistent data using special objects, files and relational databases. However, this maintenance of *persistent data*, does not disclose or suggest a *persistent process* that generates dynamic and interactive hypertext markup language (HTML) content for the application, as set out in claim 1 of the present case.

Independent Claim 12:

Independent claim 12 sets out a computer-implemented method. In step (a) of claim 12, a persistent process that generates dynamic and interactive hypertext markup language (HTML) content for an application is run. This is not disclosed by Kalajan or any of the other art cited by Examiner in the Office Action mailed March 30, 2005.

Kalajan discloses a persistent proxy socket application 56, shown in Figure 4. Persistent proxy socket application 56 handles primitives (or other messages) to and from TCP mail server 58. See Kalajan column 4, lines 12 through 14. Persistent proxy socket application 56 does not generate dynamic and interactive hypertext markup language (HTML) content for the application, as set out in claim 12 of the present application.

Kalajan is not concerned with generating HTML content. Rather, the whole thrust of Kalajan is to emulate a connection-oriented session across a network using a stateless communication protocol. See the Abstract of Kalajan. Thus, the persistent proxy socket application 56 disclosed by Kalajan is used specifically for the purpose of handling primitives (or other messages) to and from TCP mail server 58. Proxy socket application 56 remains active during an entire emulated TCP session, handling TCP message flow to and from TCP mail server 58. This allows transient proxy socket application 54 to terminate upon each output message, as required by HTTP. See Kalajan at column 4, lines 16 through 28. Nowhere does Kalajan disclose or suggest that proxy socket application 56 generates dynamic and interactive hypertext markup language (HTML) content for an application.

Examiner has asserted that Kalajan discloses that CGI program on the server could call the proxy socket applications directly to implement an HTML based email client (col. 6, lines 46 -51). However, this statement by Kalajan does not specifically state where or how HTML content is generated. In known prior art systems, HTML content is generated by transient CGI programs, such

as the CGI program mentioned by Kalajan. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan are any different than the transient CGI programs of the known prior art. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan include persistent processes. Nothing in Kalajan discloses or suggests that HTML content is generated anywhere else besides where HTML is always generated, within transient CGI programs, such as the CGI programs mentioned by Kalajan.

Hoffman's discussion of persistent *data*:

Examiner has also asserted that Hoffman, at column 2, lines 29 through 53 and column 8, lines 38 through 49, discloses the concept of generating dynamic and interactive HTML content in which dynamic HTML pages may be created using JavaScript to process user input and maintain persistent data using special objects, files, and relational databases (col. 2, lines 29-53, col. 8, lines 38-49). However, Hoffman gives no information about how HTML content is generated on a server. Rather, Hoffman concerns a method and apparatus for processing a document retrieved *from* a server. See the Abstract. The sections of Hoffman cited by Examiner (col. 2, lines 29-53, col. 8, lines 38-49) primarily discuss how a browser can interpret *client-side* JavaScript statements embedded in an HTML page received from a server. Hoffman gives no information on how the HTML page is generated.

At column 2, line 36 through 39, Hoffman indicates that using JavaScript, dynamic HTML pages may be created in which the pages process user input and

maintain persistent data using special objects, files and relational databases. However, this maintenance of persistent *data*, does not disclose or suggest a persistent *process* that generates dynamic and interactive hypertext markup language (HTML) content for an application is run, as is set out in claim 12 of the present case.

Independent Claim 21:

Independent claim 21 sets out storage media that stores a computer application. The computer application, when executed on a computing system comprises a persistent process that generates dynamic and interactive hypertext markup language (HTML) content for the computer application. This is not disclosed by Kalajan or any of the other art cited by Examiner in the Office Action mailed March 30, 2005.

Kalajan discloses a persistent proxy socket application 56, shown in Figure 4. Persistent proxy socket application 56 handles primitives (or other messages) to and from TCP mail server 58. See Kalajan column 4, lines 12 through 14. Persistent proxy socket application 56 does not generate dynamic and interactive hypertext markup language (HTML) content for the application, as set out in claim 21 of the present application.

Kalajan is not concerned with generating HTML content. Rather, the whole thrust of Kalajan is to emulate a connection-oriented session across a network using a stateless communication protocol. See the Abstract of Kalajan. Thus, the persistent proxy socket application 56 disclosed by Kalajan is used

specifically for the purpose of handling primitives (or other messages) to and from TCP mail server 58. Proxy socket application 56 remains active during an entire emulated TCP session, handling TCP message flow to and from TCP mail server 58. This allows transient proxy socket application 54 to terminate upon each output message, as required by HTTP. See Kalajan at column 4, lines 16 through 28. Nowhere does Kalajan disclose or suggest that proxy socket application 56 generates dynamic and interactive hypertext markup language (HTML) content for an application.

Examiner has asserted that Kalajan discloses that CGI program on the server could call the proxy socket applications directly to implement an HTML based email client (col. 6, lines 46 –51). However, this statement by Kalajan does not specifically state where or how HTML content is generated. In known prior art systems, HTML content is generated by transient CGI programs, such as the CGI program mentioned by Kalajan. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan are any different than the transient CGI programs of the known prior art. Nothing in Kalajan discloses or suggests that the CGI programs set out by Kalajan include persistent processes. Nothing in Kalajan discloses or suggests that HTML content is generated anywhere else besides where HTML is always generated, within transient CGI programs, such as the CGI programs mentioned by Kalajan.

Hoffman's discussion of persistent *data*:

Examiner has also asserted that Hoffman, at column 2, lines 29 through 53 and column 8, lines 38 through 49, discloses the concept of generating dynamic and interactive HTML content in which dynamic HTML pages may be created using JavaScript to process user input and maintain persistent data using special objects, files, and relational databases (col. 2, lines 29-53, col. 8, lines 38-49). However, Hoffman gives no information about how HTML content is generated on a server. Rather, Hoffman concerns a method and apparatus for processing a document retrieved *from* a server. See the Abstract. The sections of Hoffman cited by Examiner (col. 2, lines 29-53, col. 8, lines 38-49) primarily discuss how a browser can interpret *client-side* JavaScript statements embedded in an HTML page received from a server. Hoffman gives no information on how the HTML page is generated.

At column 2, line 36 through 39, Hoffman indicates that using JavaScript, dynamic HTML pages may be created in which the pages process user input and maintain persistent data using special objects, files and relational databases. However, this maintenance of persistent *data*, does not disclose or suggest a persistent *process* that generates dynamic and interactive hypertext markup language (HTML) content for the computer application, as is set out in claim 21 of the present case.

Conclusion

Applicant believes that the present case is in condition for allowance and favorable action is respectfully requested.

Respectfully submitted,

DAMIEN R. FORKNER, ET AL.

By _____
Douglas L. Weller
Reg. No. 30,506

September 7, 2005
Santa Clara, California
(408) 985-0642